RDB を用いた複合イベント処理システムの開発

金山 貴紀 † 石川 佳治 § 杉浦 健人 § 佐々木 勇和 §
† 名古屋大学工学部電気電子・情報工学科 § 名古屋大学大学院情報科学研究科

1 はじめに

ストリーム的に得られるイベントからより高次のイベントを検出する**複合イベント処理**（complex event processing, CEP）は，近年大いに着目されている [4]．複合イベントを検出するアプローチの一つは，データストリームに対する問合せ言語に，正規表現などのパターン記述機能を持たせることである [2, 6]．

一方，人工知能の分野を中心に，イベントを論理的な枠組みでとらえようとするアプローチも研究されてきており，その代表的なものが **Event Calculus** である [5]．Event Calculus についてはさまざまな研究が行われており，実際の複合イベント検出を対象とした実装技術の開発も進んでいる [1]．

本研究では [1] の Event Calculus による複合イベント処理のフレームワークを，リレーショナルデータベースの技術をもとに拡張する．[1] では，複合イベントは主記憶上で処理されることを想定しており，実装および利用における制約が大きかった．これに対し，本研究では，イベントをリレーショナルデータベース（RDB）に蓄積する．過去に検出されたイベントの問合せが可能となるだけでなく，複合イベント処理をリレーショナルデータベース管理システム（RDBMS）の問合せ処理機能を用いて実行することが可能になる．

2 Event Calculus とは

Event Calculus にはさまざまな変種がある．ここでは [1] で用いられている **RTEC** を想定する．

Event Calculus の基本概念として event と fluent がある．**event** はある時点で発生したイベントを表す．たとえば $wakeup(P)$ は，人物 P が起床したというイベントである．一方，**fluent** は時刻によって値が変わりうる性質を表す．たとえば $awake(P)$ は人が起きているかどうかを表す fluent であり，$awake(P) = \texttt{true}$ は人が起きているという状態を表す．fluent の値は，event の発生により更新されうる．

Event Calculus においては，いくつかの述語が提供されている．$\texttt{happensAt}(E, T)$ は，event E が時刻 T において生じたことを示す．なお，Event Calculus は離散的な時間を扱っている．$\texttt{holdsAt}(F = V, T)$ は，時刻 T における fluent F の値が V であることを示す．$\texttt{holdsFor}(F = V, I)$ は，fluent F の値が V であるような時区間の集合で最大となるものが I であることを示す．I が時区間のリストであることに注意する．$\texttt{initiatedAt}(F = V, T)$ は，fluent F の値が時刻 T において V になったことを示す．逆に，$\texttt{terminatedAt}(F = V, T)$ は，fluent F の値が時刻 T において V でなくなることを示す．

イベント記述（event description）はルールの形式で与えられる．以下のルール（R_1 とする）は [1] からの例であり，交通流のストリーム S において，1 分間で混雑度が 1.2 倍より大きくなったときに，fluent $densityTrend(S)$ が増加状態に入った（$increasing$）と判断するルールである．

$$\begin{aligned}&\texttt{initiatedAt}(densityTrend(S) = increasing, T) \leftarrow\\ &\quad \texttt{happensAt}(traffic(S, Flow, Density), T),\\ &\quad \texttt{happensAt}(traffic(S, Flow', Density'), T + 60),\\ &\quad Density' > Density \times 1.2\end{aligned}$$

以下のルール（R_2 とする）は，一緒に歩いていた 2 名の人と，その歩いていた時区間の集合を検出するルールである．

$$\begin{aligned}&\texttt{holdsFor}(moving(P_1, P_2) = \texttt{true}, I) \leftarrow\\ &\quad \texttt{holdsFor}(walking(P_1) = \texttt{true}, I_1),\\ &\quad \texttt{holdsFor}(walking(P_2) = \texttt{true}, I_2),\\ &\quad \texttt{holdsFor}(close(P_1, P_2) = \texttt{true}, I_3),\\ &\quad \texttt{intersect\_all}([I_1, I_2, I_3], I)\end{aligned}$$

2，3 行目において人が歩いていたという fluent $walking$ が，また，4 行目において 2 人の人が近くにいたという fluent $close$ が用いられている．これらの持続時間 I_1, I_2, I_3 はさらに別のルールで導かれる．

3 システムのアーキテクチャ

想定するシステムアーキテクチャを図 1 に示す．入力は **SE ストリーム**である，SE は複合イベントでない**単純イベント**（simple event）を意味している．先に示した例では，時刻 T における交通流の状態 $traffic(\ldots)$ を示すイベントが対応する．**イベント記述**（event description）は，システムに事前にユーザや管理者などにより与えられている複合イベントの導出のためのルールである．**CEP マネジャ**（CEP manager）は，与えられたイベント記述をもとに，**フィルタ**（filter）を導出する．これは，SE ストリームから必要なものだけを抽出するために利用される．

複合イベントの検出処理を担当するのが **CEP エンジン**（CEP engine）である．CEP マネジャから与えら

Development of Complex Event Processing System Based on RDB
Atsuki Kanayama †, Yoshiharu Ishikawa §, Kento Sugiura §, Yuya Sasaki §
† Department of Information Engineering, School of Engineering, Nagoya University
§ Graduate School of Information Science, Nagoya University

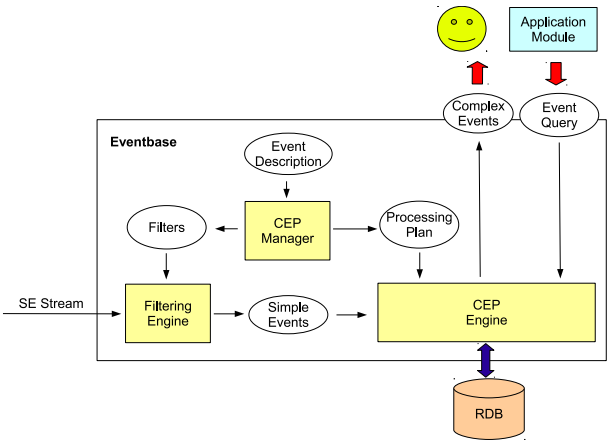

図 1: システムアーキテクチャ

れた処理プランをもとに処理を行う．検出された複合イベントは，ユーザおよびアプリケーションモジュールに報告される．一方，CEP エンジンはアドホックな問合せ機能も提供しており，複合イベントに対する問合せを直接受け取り，処理することもできる．

4 複合イベント処理のアプローチ

本節では，Event Calculus に基づく複合イベント処理において，RDBMS を効果的に用いる方式について例を用いて説明する．

例：ルール R_1 について 基本的なアイデアは，リレーションを用いてイベント処理を表現することにある．例として，先に示した fluent *densityTrend* に関するルールを考える．このルールの前提として，交通流に対する単純イベントのストリームに対応する

```
SE@traffic(S, Flow, Density, T)
```

というリレーションがシステム上に存在するものとする．このリレーションには，交通流に関する単純イベントが到着するごとに追加されていく．一方，ルールの頭部に対応する，fluent の変化を記録するためのリレーション

```
CE@densityTrend(S, Pred, Value, I)
```

を作成する．`SE@traffic` リレーションに対して

```
SELECT x.S, "initiatedAt", "increasing", [x.T, x.T]
FROM SE@traffic x, SE@traffic y
WHERE y.T = x.T+60 AND y.Density > x.Density*1.2
```

という問合せを実行し，*densityTrend* が増加に移った時刻に対するタプルを生成・挿入する *．

例：ルール R_2 について 次いで，第 2 のルールについて考える．ここでは，ルールの本体部の

```
CE@walking(P, Pred, Value, I)
CE@close(P, Pred, Value, I)
```

という 2 つのリレーションがすでに定義されていることが前提となる．`I` は開始時刻と終了時刻のペアからなる時区間データである †．最近の RDBMS では時区間データ型，もしくはユーザ定義型の機能が存在するため，これらを用いて対応する．一方で，ルールの頭部に対応するリレーション

```
CE@moving(P1, P2, Pred, Value, I)
```

を定義する．これらにより，R_2 に対応する問合せは

```
SELECT x.P, y.P, "holdsFor", "true",
       intersect_all([x.I, y.I, z.I])
FROM CE@walking x, CE@walking y, CE@close z
WHERE x.Value = true AND y.Value = true AND
      z.Value = true AND
      is_intersect_all([x.I, y.I, z.I])
```

となる．ここで `is_intersect_all` は，引数として与えられる時区間のリストに交わりがあるときに真になるユーザ定義述語であり，`intersect_all` は，交わりとして得られる時区間を返すユーザ定義関数である．

実際の問合せ処理においては，新たな SDE の到着に合わせてインクリメンタルにリレーションを更新していく必要がある．このためには，多くの研究がある実体化ビューのインクリメンタルな更新に関する手法 [3] が活用できると考えている．

5 まとめと今後の課題

本稿では Event Calculus に基づく複合イベント処理を RDB 技術を用いて実現するアプローチについて述べた．今後は，システムとして提供する機能について具体化を進め，実装方式に関する開発を行う．具体的に RDBMS 上での実装を図る．

謝辞

本研究は，JST COI（Center of Innovation）プログラムおよび科研費（25280039, 26540043）による．

参考文献

[1] A. Artikis, M. Sergot, and G. Paliouras. An event calculus for event recognition. *IEEE TKDE*, 27(4):895–908, 2015.

[2] H.-L. Bui. Survey and comparison of event query languages using practical examples. Graduate Thesis, Institut für Informatic, Ludwig-maximilians-Universität München, Mar. 2009.

[3] S. Ceri and J. Widom. Deriving production rules for incremental view maintenance. In *VLDB*, pp. 577–589, 1991.

[4] G. Cugola and A. Margara. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys*, 44(3), 2012.

[5] Wikipedia: Event calculus. http://en.wikipedia.org/wiki/Event_calculus.

[6] E. Wu, Y. Diao, and S. Rizvi. High-performance complex event processing over streams. In *ACM SIGMOD*, pp. 407–418, 2006.

*時刻 `x.T` を，開始時刻と終了時刻が同じ時区間 `[x.T, x.T]` として表現している．

†元の定義では I は時区間のリストであるが，実装では各時区間ごとに異なるタプルに分ける．